

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problems Mailbox.**

THIS PAGE BLANK (USPTO)

Disclosure Publication
DE 197 40 550 A 1
Int. Cl. G 05 B 19/04

File number: 197 40 550.9
Application Date: 9/15/97
Disclosure Date: 4/16/98

Internal Priority: 296 17 837.3 10/14/96

Applicant: Siemens AG, 80333 Munich, Germany

Inventors:
Heber, Dr. Tino, 09599 Freiberg, Germany; Kirste, Dr. Steffen, 09120 Chemnitz; Hess, Prof. Karl, 09122 Chemnitz; Wucherer, Klaus, Cert. Eng., 90610 Winkelhaid, Germany.

The following information is taken from documents submitted by the applicant

Controller

The invention is a controller equipped with the means to control a technical process and/or with the means to control the movement of a processing machine; a control program can be supplied for this controller to process the control during operation. The implementation of process functionalities and technological moving processes for processing machines is facilitated by the fact that the control program is provided with software modules processed by at least one CPU unit during operation, while the software modules are configured in such a way that they serve the purpose of process control and/or movement control. The invention is used for SPS/NC controls.

Illustration:

BB

Bu ST

Multiple-axis module 0 Multiple-axis module 1 Multiple-axis module 2 Single-axis module 3 Single-axis module 4 Single-axis module 5
St0 St1 St2 St3 St4 St5

Input/Output (digital, analogue) Ae Ae Ae Ae Ae Ae

master axis left gripper right gripper shaft pile thread shaft filling thread shaft binding thread pile wire feed pile wire removal pile wire transvers
transport needle roller
pile thread storage filling thread storage binding thread storage textile storage

Description

The invention concerns a controller equipped with the means to control a technical process and/or with the means to control the movement of a processing machine; a control program can be supplied for this controller and processed by the controller during the control operation. In addition, the invention concerns a programming device with the means to create a control program for a controller of this type.

A programmable controller and a programming device to create a control program for a programmable controller of this type are described in the Siemens catalogue ST 70, 1996 edition, Chapters 3, 4, and 8. The principal components of this programmable controller are assemblies for central tasks (CPU units) and signal, function, and communication assemblies. The CPU unit of the programmable controller processes a control program cyclically during operation; the control program is created by a programmer using a programming device equipped with a software tool, and is intended for the solution of a task to be automated. During cyclical processing, the CPU unit first reads the signal status for all physical process inputs and creates a process map of the inputs. The control program is processed in single-step execution with the aid of internal counters, markers, and times, and finally the CPU unit stores the calculated signal statuses in the process map of the process outputs; the signal statuses are forwarded from the process map to the physical process outputs. This control program usually includes software functional modules that enable the operation of the signal and/or function and/or communication assemblies. One of these function assemblies, in the form of an NC controller assembly, can be used to control the movement processes of a processing machine. For this purpose, the CPU unit which typically realizes process controller functionalities transmits parameters to this NC controller assembly, in the form of start/stop coordinates for the processing machine's drive axes that are to be controlled, for example. Furthermore, the CPU unit selects executable procedural programs on the NC controller assembly which are executed by a processor of the NC controller assembly for control of the movement process of a processing machine.

The objective of the present invention is to describe a controller of the type mentioned above that simplifies the realization of the process functionalities and technological movement processes of processing machines.

In addition, a programming device must be created that simplifies the creation of a control program for a controller of this type.

This task is solved in respect to the controller by the steps indicated in the pertinent part of Claim 1, and in respect to the programming device by the steps indicated in the pertinent part of Claim 6.

It is advantageous that the process control functionalities of known programmable controllers (SPSs) and the movement functionalities of known NC controllers or NC controller assemblies in a single uniform, configurable controller system. In this way, project-specific controllers can be constructed as variants in one configuration phase, and the need to combine separately available "SPS technology" and "NC technology" into a single system is avoided.

Advantageous realizations for the invention result from the steps described in the sub-claims

With reference to the drawing showing a sample configuration for the invention, the following presents a more detailed explanation of the invention, its realizations, and its advantages.

Fig. 1 shows the program structure of a software module.

Fig. 2a to 4b show the declaration tables,

Fig. 5a to 7b show the movement command tables,

Fig. 8 shows a declaration table for axis combinations,

Fig. 9 shows a profile declaration table,

Fig. 10 shows a movement attribute table.

Fig. 11 shows a movement function table,

Fig. 12 shows a configuration element table,

Fig. 13 shows a variable declaration table,

Fig. 14 shows an access path declaration table,

Fig. 15 shows a communication function table,

Fig. 16 shows the basic structure of a pile wire mechanical loom,

Fig. 17a and 17b show a movement diagram for a pile wire mechanical loom, and

Fig. 19 shows a controller structure.

In Fig. 1, the module identified as 1 is intended to realize the movement process of a processing machine; it is created by a programmer on a programming device not shown here. The module 1 is part of a control program that can be transferred online or offline into a controller after it has been translated into the controller's appropriate machine language; the module is processed by a CPU unit of this controller during operation. The module 1 is composed of a declaration part 2, at least one cyclical program 3a, 3b, and at least one sequential program 4a, 4b. All the programs 3a, 3b, 4a, 4b of the module 1 access the declaration part 2, and program names, program types, variables, and/or data structures and/or movement profiles are stored in this declaration part 2. The cyclical programs 3a, 3b are intended to coordinate the sequential programs 4a, 4b that can be called up by these programs 3a, 3b. If modules for process control are provided, the cyclical programs of such modules implement the functionalities of a programmable controller. Regardless of whether the modules serve to implement process functionalities and/or to implement the movement functionalities of a processing machine, the CPU unit of the controller processes these modules. Within this module 1, local variables, input and output variables, and sequential and cyclical programs are programmed, configured, and declared with a programming device. The programs that belong to the module have unrestricted access to all of a module's variables. Declaration regulations for the modules and their variables are provided for this purpose. Examples of such declaration regulations are shown in Fig. 2a, 2b, 3, and 4, where Tables 1 to 4 present a declaration of modules and of keywords for the variables, examples of a variables declaration, and an assignment of priority for variables.

The cyclical programs 3a, 3b contain language resources with suitable instructions and commands for starting sequential programs and providing functional modules with parameters. Specifically, the following particular language elements are available within a program for the cyclical process:

- operators, such as comparative or binary operators,

- location functions, such as type conversion functions for elementary data types, mathematical functions, binary functions, and functions for accessing system variables,
- standard functional modules, such as functional modules for signal edge recognition, bistable functional modules, or counter and time modules, and
- instruction elements in the form of selection, repetition, and go-to statements, and in the form of control instructions for functions, functional modules, and programs.

Each sequential program 4a, 4b corresponds to a non-periodical task. The priority of the task is allocated to a sequential program within the declaration. Sequential programs are started by other programs and provide feedback values when called up, with which they are managed within the system (for example, locking to prevent repeated call-up). A module can feature no sequential program, one sequential program, or several sequential programs. All movement functionalities are available only in sequential programs. Thus a sequential program contains the scope of commands covering all movement commands. In addition, a sequential program can also feature commands for logical processing. Fig. 5a, 5b, 6, 7a, and 7b show examples of movement functionalities, while Table 5 presents general movement commands, Table 6 interpolation movements, and Table 7 movement commands for a master-slave group.

Each of the cyclical and sequential programs 3a, 3b, 4, 4b includes a variables and constants declaration part 5, in which user-specific variables and constants are to be declared. In particular, the following are declared:

- declaration of local variables with elementary data types, for example, whole-number or real data types, strings,
- definition of derived data structures and movement profiles,
- declaration of system variables (axis handle),
- allocation of variables to logical device addresses,
- assignment of access privileges for variables made available for data exchange,
- multiple-axis configuration by declaration of different axis combinations (Fig. 8),
- definition of movement profiles (Fig. 9)

Tables 8 and 9 in Fig. 8 and 9 represent examples of a declaration of axis connections (multiple-axis configuration) and of a declaration of movement profiles.

In addition to the declaration of variables and constants, a declaration of functional modules is provided. It has been implicitly defined for the functional modules whether they need a fast cyclical task when called up or whether they can be fitted to the context of the program calling them up. Functional modules that run in the context of the calling program are instantiated within that program. The number and instance names of fast functional modules are fixed values within the controller system. Functional modules are executed periodically and can be provided with new parameters. The execution of fast functional modules is not part of the responsibility of the controller for the calling task. Thus, execution is independent of the analysis rules of the program in which the functional module was parameterized. All other functional modules run in the context of the calling program, in other words, they follow the sequence of analysis of the program's language elements

In particular, the following language elements are provided for the implementation of movement functionalities:

- standard functional modules (for example, cam feeding mechanism),
- mechanisms for multiple-axis configurations (using axis modules for configuration of very different axis combinations into a complete system),
- movement-specific expanded (derived) data structures,
- movement attributes, functions, and commands.

Tables 10 and 11 in Fig. 10 and 11 present examples of principal movement attributes and movement functions.

Configuration elements can be preset for the configuration by means of axis modules of a wide range of different axis combinations into a controller for the control of a technical process and/or for the control of the movement of a processing machine. These elements include:

- resources in the form of hardware resources,
- modules,
- global variables,
- access paths,

where a declaration of resources, a declaration of global variables for the conjoining of modules with different resources, and a declaration of access paths can be preset within a configuration. Tables 12 to 14 in Figs. 12 to 14 present configuration elements, a declaration of global variables, and a declaration of access paths. Global variables for the conjoining of modules within a resource and module are declared within the resource itself. An access path is provided for the conjoining of a variable with an input or output variable in a module, for the conjoining of a variable with global variables in a resource or configuration, or for the conjoining of a variable with a directly represented variable. In addition to a declaration of global variables for data exchange between modules and programs (of one or various resources), data can be exchanged via functional modules. Table 15 in Fig. 15 presents examples of communication functions.

The plan of a configurable controller is explained in the following. Refer to Fig. 16, which depicts the basic construction of a pile wire mechanical loom suitable to the manufacture of Wilton and Boucle carpets. The principal components of this mechanical loom are a batten 6, a gripper pair 7 for feeding the filling thread, a dobby, a pile wire mechanism 9, a warp thread and pile thread storage 10, a textile output 11, and a textile storage 12.

In determining the inputs, a fundamental distinction is drawn between time-critical and non-time-critical inputs. Time-critical inputs include monitor signals (for example, filling stop movements, pile wire monitors, stop signals) that require the controller to react in the lowest time level (IPO cycle). Signals that activate the emergency off function of the controller (emergency off switch, drive monitoring) are processed separately. The remaining input signals, such as operating actions and non-time-critical monitors (textile output, textile storage, etc.), are processed in the main cycle of the corresponding module.

In determining statuses, a fundamental distinction is drawn between the following operating conditions of the machine:

- 1) JOG – free operation of the axes/drive assemblies according to the operator's selection
- 2) JOG-Reference – reference of the axes according to the operator's selection or the appropriate preset,
- 3) AUTOMATIC (programmed service):
 - stationary operation (weaving),
 - routines for handling exceptional circumstances due to the machine or the process.

A technological movement process must be set by a user for stationary operation, for example, a movement process as depicted in Fig. 17a and 17b:

1. open shed 1
 - a) move the heald shafts into the resting position for the first shoot, and the batten into the back end position;
2. feed in the filling thread and pile wire:
 - a) movement of the gripper bars into the shed,
 - b) transfer of the filling thread from the left to the right gripper bar,
 - c) return movement of the gripper bars,
 - d) feed in the pile wire into the upper part of the shed;
3. controlling the cutting/clamping device
 - a) cutting off of the filling thread, and fixing it until the next filling thread feed;
4. close the shed, crimp off filling thread and pile wire:
 - a) movement of the heald shafts into the intermediate position.
 - b) move the batten into the front end position for crimping the filling thread and pile wire,
 - c) repositioning the pile wire feed;
5. open shed 2:
 - a) movement of the heald shafts into the resting position for the second shoot, and the batten into the back end position;
6. feed in filling thread;
7. controlling the cutting/clamping device;
8. close the shed, crimp off the filling thread;
9. continue in the cycle (1).

Additional movement processes are to be realized parallel to the basic cycle:

1. pile wire removal:
 - a) removing the last pile wire before textile is output and pushing it into a pile wire box;
2. transverse transport of the pile wire:
 - a) transverse transport of the pile wire between the movements of pile wire feed and pile wire removal (maintaining the pile wire cycle);
3. Textile output.
 - a) continuously running needle roller for producing textile;
4. supply of warp and pile threads:

- a) continuous supply of two warp thread systems and one pile thread system;
 5. taking up the textile:
 a) activation of the finished textile storage.

In addition, the user presets the movement functionalities of the individual axes/drive assemblies and the nature of output quantities and other physical quantities in respect to a so-called main shaft. For the present example, the following output and movement functionalities are preset:

Axis/Drive Assembly or Output Quantity	- Description	- Parameters
Main shaft	<ul style="list-style-type: none"> - continuously running round axis - system master axis 	<ul style="list-style-type: none"> - rotations main shaft
Batten	<ul style="list-style-type: none"> - mechanically coupled to the main shaft - movement function is realized mechanically 	<ul style="list-style-type: none"> - none
Left Gripper	<ul style="list-style-type: none"> - movement function in accordance with VDI directive 2143 for radial cams - 9th-degree polynomial 	<ul style="list-style-type: none"> - gripper path - zero point - angle of the main shaft
Right Gripper	<ul style="list-style-type: none"> - left gripper 	<ul style="list-style-type: none"> - left gripper
Cutting/Clamping Device	<ul style="list-style-type: none"> - digital output signal for controlling the pneumatic cutting/clamping device - determined by the angular position of the main shaft 	<ul style="list-style-type: none"> - angle - main shaft for H and L signal
Shaft 1, Pile Thread	<ul style="list-style-type: none"> - movement function in accordance with VDI directive 2143 for radial cams - 3rd-degree polynomial 	<ul style="list-style-type: none"> - shaft path - zero point - angle of the main shaft
Shaft 2, Filling Thread	<ul style="list-style-type: none"> - shaft 1 	<ul style="list-style-type: none"> - shaft 1
Shaft 3, Binding Thread	<ul style="list-style-type: none"> - shaft 1 	<ul style="list-style-type: none"> - shaft 1
Storage Pile Thread	<ul style="list-style-type: none"> - continuous unwinding of the thread storage during main shaft movement - rotation speed is leveled off between limit initiators 	<ul style="list-style-type: none"> - thread tension (limit initiators) - motor revolutions
Storage Filling Thread	<ul style="list-style-type: none"> - unwinding of the 	<ul style="list-style-type: none"> - thread tension (limit

	storage at maximum thread tension until minimum thread tension is achieved - drive assembly controlled by start/stop signal at fixed pre-set revolutions	initiators)
Storage Binding Thread	- Storage Filling Thread	- thread tension (limit initiators)
Needle Roller	- continuous revolution in relation to the main shaft	- textile density (technological parameter)
Textile Storage	- revolution from minimum textile tension until maximal textile tension has been achieved - drive assembly controlled by start/stop signal at fixed pre-set revolutions	- textile tension in the finished goods storage (limit initiators)
Pile Wire Feed	- movement appropriate to the angulations preset for the main shaft - trapezoid profile	- none
Pile Wire Removal	- movement for removal at a constant speed corresponding to the pre-set angular range of the main shaft - **?*?*	- speed and acceleration (thread clamping)
Transverse Transport of Pile Wire	- movement corresponding to the angulations preset for the main shaft - trapezoid profile	- none

The programmer configures the software modules of the control program in accordance with the pre-set technological movement process, the pre-determined movement functionalities of the axes/drive assemblies, and the status of output and other physical quantities; in the present example, several CPU units are provided for the purpose of processing the modules during operation. The following modules are configured in the example:

1. Multiple-axis module 0: main shaft and gripper mechanism

a) operating mode control

ADJUST – routines for handling exceptional situations due to processes or machines,

STATIC – stationary operation, “weaving”

- b) evaluation and implementation of operating requirements,
- c) logical connection of the inputs and outputs required for the process,
- d) programs for describing the movement of the connected axes (main shaft and gripper mechanism),
- e) activation of the required axis combinations or single-axis movements in other modules,
- f) monitoring of machine and process status,
- g) handling system errors;
- 2. Multiple-axis module 1: dobby
 - a) evaluation and implementation of command requirements for the multiple-axis module 0,
 - b) program for describing the movement of the connected axes (dobby);
- 3. Multiple-axis module 2: pile wire device
 - a) evaluation and implementation of command requirements for the multiple-axis module 0,
 - b) program for describing the movement of the connected axes (pile wire device),
 - c) monitoring of the process status of the subsystem;
- 4. Single-axis module 3: needle roller
 - a) the module does not contain any separate programs,
 - b) it is located in the “non-cyclical command operation” operating mode and thus has a command interface to the multiple-axis module 0,
 - c) the module receives the commands for movement of the drive assembly, with indications of revolutions and rotational direction, via this interface;
- 5. Single-axis module 4: pile thread storage
 - a) the module contains the program for controlling the pile thread storage,
 - b) evaluation and implementation of the command requirements for the multiple-axis module 0,
 - c) logical connection of the inputs and outputs required for the process,
 - d) monitoring of the process status of the subsystem;
- 6. Input/output module 5: Filling and binding warp storage
 - a) the module contains a separate program to control the filling and binding warp drive assemblies (drive assemblies are controlled by start/stop signals; number of revs is defined in the drive assemblies),
 - b) logical connection of the inputs and outputs required for the process,
 - c) monitoring of the process status of the subsystem.

The following refers to Fig. 18, which depicts a controller structure for processing the modules. In the example, the controller ST comprises six sub-controllers St0 ... St5, each of which is provided with a CPU unit; they are connected to each other via a suitable bus Bu. The CPU unit of the sub-controller St0 processes the multiple-axis module 0; the CPU unit of the sub-controller St1 processes the multiple-axis module 1. The CPU unit of the sub-controller St2 processes the multiple-axis module 2, the CPU unit of St3 the single-axis module 3, the CPU unit of St4 the single-axis module 4, and the CPU unit of St5 the input/output module 5. Drive assemblies with suitable drive axes are connected to the sub-controllers St0 ... St5 via suitable output units Ae; the drive axes are in

mechanical linkage with each other in accordance with the constraints of the control program that includes the software modules. An operating and monitoring station BB is provided for operating and observing the technical process and/or the movement process of the pile wire loom.

Patent Claims.

1. Controller which is equipped with the means to control a technical process and/or with the means to control the movement of a processing machine, and which will be processed by the control program during control operation, **distinguished by the fact that the control program is provided with software modules which are processed during operation by at least one CPU unit of the controller; the software modules are configured in such a way that they are provided to control a process and/or movement.**
2. Controller as in Claim 1, distinguished by the fact that
 - the number of drive axes that can be connected to the input/output units of the controller and the combined actions of those axes are determined by the technological movement process of the processing machine, and
 - single-axis and multiple-axis modules have been configured in accordance with the pre-determined number of drive axes and their combined actions for controlling movement.
3. Controller as in Claim 1 or 2, distinguished by the fact that the software modules feature at least one cyclical program and at least one sequential program that can be called up by the cyclical program, where
 - in the case of movement control, the sequential program is provided for realizing the movement functions and the cyclical program is provided for coordinating the sequential programs, and
 - in the case of process control, the cyclical program is provided for implementing process control functionalities.
4. Controller as in Claim 3, distinguished by the fact that each module is provided with a declarations part accessed by the programs of a given module and in which variables and/or data structures and/or movement profiles are stored.
5. Controller as in Claim 3 or 4, distinguished by the fact that
 - a program is furnished with at least one functional module, and
 - functional modules can be called up by a program.
6. Programming device with the means to create a control program for a controller that includes the means to control a technical process and/or the means to control the movement of a processing machine, distinguished by the fact that those means provide the control program with software modules which are processed by a CPU unit of the controller during operation, where the software modules are configured in such a way that they are provided for process control and/or movement control.
7. Programming device as in Claim 6, distinguished by the fact that
 - the number of drive axes that can be connected to the input/output units of the controller and the combined actions of those axes can be determined by the technological movement process of the processing machine, and

- single-axis and multiple-axis modules can be configured in accordance with the pre-determined number of drive axes and their combined actions for controlling movement.
- 8. Programming device as in Claim 6 or 7, distinguished by the fact that the means provide at least one software module with at least one cyclical program and with at least one sequential program that can be called up by the cyclical program, where
 - in the case of a movement control, the sequential program is provided for implementing the movement functions and the cyclical program is provided for coordinating the sequential programs, and
 - in the case of a process control, the cyclical program is provided for implementing process control functionalities.
- 9. Programming device as in Claim 8, distinguished by the fact that each module is provided with a declarations part accessed by the programs of a given module and in which variables and/or data structures and/or movement profiles are stored.
- 10. Programming device as in Claim 8 or 9, distinguished by the fact that
 - a program is provided with at least one functional module, and
 - functional modules can be called up by a program.
- 11. Arrangement with at least one controller as in one of the Claims 1 through 5 and with at least one programming device as in one of the Claims 6 through 10, whereby the controller and the programming device are connected to each other by a bus.

20 page(s) of drawings

DRAWINGS, Page 1

Number: DE 197 40 550 A1

Int. Cl.⁸ G 05 B 19/04

Date of publication: April 16, 1998

FIG 1

[upper box:]

```

VAR_INPUT
  start, stop: BOOL;
  status: UINT;
END_VAR
VAR_OUTPUT
  ges_status: UINT;
END_VAR
VAR (* declaration of module-global variables *)
  master: AXIS := Axis0;
  slave: AXIS := Axis1;
  master_field: ARRAY [0..11] OF REAL := 0,30,60,90,120,...,330;
  slave_field: ARRAY [0..11] OF REAL := 0,0,0,10,20,...,0,17,5;

  conjunction: DEFCAM := master, slave;
  mprofil: TPROFIL(master_field)
  sprofil1: TPROFIL(slave_field1, 0.2);
  sprofil2: TPROFIL(F3(5.32, 0.45, ), 0.2);

```

[lower left box:]

PROGRAM *process* (TYPE:=NORM, PRIORITY:=2)*Declarations part*

VAR (* local variables *)

```

  nulip, change: BOOL;
  erg: REAL;

```

END_VAR

Process part

```

IF((start=1) AND (nulip = 1)
  THEN CREATE (default)

```

END_IF

```

IF((change=1) AND (start=1)
  THEN CREATE (pchange)

```

```

ELSIF (ges_status <= 0) THEN stop:=1;

```

END_PROGRAM

PROGRAM *process* (TYPE:=FAST, INTERVAL:=2)

END_PROGRAM

[lower right box:]

PROGRAM *default* (TYPE:=SEQ, PRIORITY:=2)*Declarations part*

VAR

```

  moving: BOOL;
  toutren: UINT:=0;

```

END_VAR

Sequence

SET (slave, Profile, sprofil1);

(* MOVE master starts axis combination *)

MOVE (*combination, master, N(100)*);
WAIT FOR (*stop=1*);
STOP (*combination, master, A(0)*);

END_PROGRAM

PROGRAM *pchange* (TYPE:=SEQ, PRIORITY:=2)
 Sequence

SET (*master, profile, mprofil*);
SET (*slave, profile, sprofil2*);
MOVE (*combination, master, N(100)*);
WAIT FOR (*stop=1*);
STOP (*combination*);

END_PROGRAM

PROGRAM *name* (TYPE:=SEQ, PRIORITY:=4).

END_PROGRAM

DRAWINGS, Page 2

Number: DE 197 40 550 A1

Int. Cl.⁶: G 05 B 19/04

Date of publication: April 16, 1998

FIG 2a

Declaration Direction		Declaration	Remarks / References
Module		MODULE name: <i>Module_Designation...</i> (* module master *) END_MODULE	- the identification symbol ON is used at the logical level to determine the module type (<i>Module_Designation</i>)
Variables	local variables	VAR ... END_VAR	- local variables on the module are global for all associated programs
	input variables	VAR INPUT ...END_VAR	
	output variables	VAR_OUTPUT ...END_VA R	
Program	general declaration	PROGRAM <i>name</i> (<i>TYPE:=type</i> , <i>PRIORITY:=value</i> , <i>INTERVAL:=period</i> , <i>SYSSTART:=start type</i>) ...(* program master *) END_Program	- TYPE indicates the type of program or associated task: <ul style="list-style-type: none"> • NORM = periodic (cyclical) task • FAST = fast cyclical task • SEQ = sequential (not periodic) task - PRIORITY determines the priority for calling up the task, whether priority or not. priority (<i>valuetype</i> : UINT (0,1,...5)). programs are called up for periodic execution at an indicated INTERVAL (<i>time</i> <i>period</i>) (<i>time period</i> type INT corresponds to a multiple of the interpolation task) the SYSTART parameter is permitted only for cyclical programs, and determines whether programs are started by explicit call-up (<i>SYSSTART:=USER</i>) or by initializing the module (<i>SYSSTART:=INIT</i>)

			(USER is set as the default)
	cyclical program (without fixed time grid)	PROGRAM <i>name</i> (TYPE:=NORM, PRIORITY:= <i>value</i> , SYSSTART:= <i>start type</i>) ... (* program master *) END PROGRAM	- the program with the highest priority and with SYSTART:=INIT becomes the main entry point for the module
	fast cyclical program	PROGRAM <i>name</i> (TYPE:=FAST, INTERVAL:= <i>time period</i> , SYSSTART:= <i>start type</i>) ... (* program master *) END PROGRAM	- a maximum of one FAST-type cyclical program can be programmed in each module
	program with sequential processing	PROGRAM <i>name</i> (TYPE:=SEQ, PRIORITY:= <i>value</i>) ... (* program master *) END PROGRAM	- sequential programs are started exclusively by an explicit direction (CREATE...)

Table 1: Declaration of Modules (cont.)

FIG 2b

Drawings, page 3

Declaration	Keyword	Application / Remarks
local variables	VAR	- use within the program organization unit
input variables (write-protected)	VAR_INPUT	- provided from outside, cannot be changed in the program organization unit
input variables	VAR_IN_OUT	- variables can be changed in the program
output variables	VAR_OUTPUT	- variables provided to the outside by the program organization unit
constants	CONSTANT	- constants (cannot be changed) - declaration requires value assignment
storage location assignment	AT	- if this keyword is not given, the variables are automatically assigned to a storage location
end of the variables declaration	VAR_END	- every declaration of variables (regardless of its type) concludes with VAR_END
buffered variables	RETAIN	- during a warm start, the variables assume their buffered values - during a cold start, the variables assume the indicated values or the initialization values pre-set in the system
global variables	VAR_GLOBAL	- if global variables are declared within a configuration element, the variables' scope of validity is limited to the element in which they were defined
access path for variables	VAR_ACCESS	- establishes variables that can be accessed by communication services

Table 2: Keywords for a declaration of variables

FIG 3

Example	Remarks
VAR <i>Bit</i> : ARRAY [0..6] OF BOOL:=1,1,0,0,0,1,0; END VAR	- allocates initial values to 8 memory bits: <i>Bit</i> [0]:=1, ..., <i>Bit</i> [7]:=0
VAR <i>master</i> : INT_AXIS:=log.axis address; <i>slave</i> : AXIS:=log.axis address; END VAR	- declaration of an axis handle requires assignment to the axis's logical address
VAR AT %QX5.1: BOOL:=1; END VAR	- boolean variables, directly addressed and initialized with initial value=1
VAR <i>number, value</i> : INT; <i>mystring</i> : STRING(10); END VAR	- several variables of the same type separated by a comma - character series with a maximum length of 10
VAR CONSTANT <i>value</i> : INT:=103; END VAR	- variables with a constant value - declaration of constants requires simultaneous value assignment
VAR RETAIN <i>status</i> : ARRAY [0..3] OF INT:=1,5,0,0; END VAR	- declared as a buffered field with the initial values for cold start <i>status</i> [0]:=1, <i>status</i> [1]:=5 <i>status</i> [2]:=0, <i>status</i> [3]:=1

Table 3a: Examples of a declaration of variables

FIG 4a

Meaning	Command	Example
Communication priority in the event of simultaneous access (0-5, 0 highest priority, 3 set as default) Priority provided only for variables with data exchange	% Priority (not IEC 1131)	VAR_INPUT stop: BOOL %0; number: INT %5; END_VAR

Table 3b: Assignment of priority

Fig 4b

Movement		Command	Remarks
Referring	Single-axis system	REF	- different referring modes can be set via system variables
	Multiple-axis system	REF <i>Axis index</i>	- simultaneous referring of all axes
Positioning movement	speed-controlled	POS (<i>TYPE_{opt}</i> (<i>position</i>), <i>speed_{opt}</i>)	- single-axis system - speed from a system variable - <i>TYPE</i> : position attribute
(Positioning movement cont.)	(speed-controlled)	POS (<i>Axis index₁</i> , <i>TYPE_{opt}</i> (<i>position</i>), <i>speed_{opt}</i>) ... (<i>Axis index_n</i> , <i>TYPE_{opt}</i> (<i>position</i>), <i>speed_{opt}</i>)	- multiple-axis system - axis movements programmed within one movement command start simultaneously
(Positioning movement cont.)	(speed-controlled)	POS (<i>combination name</i> , <i>Axis index₁</i> , <i>TYPE_{opt}</i> (<i>position</i>), <i>speed_{opt}</i>)	- multiple-axis system - driving a combination within the position range of the master axis

Table 5: General Movement Commands – Single-axis and Combination

FIG 5a

	time-controlled	<p><u>POST(<i>TYPE_{opt}</i>(<i>position</i>), (<i>time</i>))</u></p> <p>POST(<i>Axis index</i>, <i>TYPE_{opt}</i>(<i>position</i>), <i>time</i>, ... <i>Axis index_n</i>, <i>TYPE_{opt}</i>(<i>position</i>), <i>time</i>)</p>	<p>- single-axis system</p> <p>- <i>time</i> indicates the duration of positioning movement</p> <p>- multiple-axis system</p>
continuous movement	single-axis movement	<p>MOVE(<i>TYPE_{opt}</i>(<i>speed</i>), ...)</p> <p>MOVE(<i>Axis index</i>, <i>TYPE_{opt}</i>(<i>speed</i>), ... <i>Axis index_n</i>, <i>TYPE_{opt}</i>(<i>speed</i>), ...)</p>	<p>- single-axis system</p> <p>- type: direction attribute</p> <p>- multiple-axis system</p> <p>- when movement is started and <i>speed</i> is attained, program processing is continued.</p>
	movement in combination	MOVE(<i>name of combination</i> , <i>Axis index</i> , <i>TYPE_{opt}</i> (<i>speed</i>), ...)	- only one axis programmable; represents the master for that combination
	combination movement according to an external master axis	MOVE(<i>name of combination</i> , ...)	<p>- <i>axis index</i> must be an external axis</p> <p>- the combination waits for the movement of the external axis (<i>axis index</i>), which it then follows immediately</p>
Axis Stop	single-axis system	STOP	
	multiple-axis system for single axis	STOP (<i>Axis index</i> , <i>Axis index_n</i>)	
	multiple-axis system for combination	STOP (<i>name of combination</i> , ...)	- immediately stops the axis combination with that <i>name of combination</i>
		STOP (<i>name of combination</i> , <i>axis index</i> , <i>position</i> , ...)	- stops the axis combination with that <i>name of combination</i> when the indicated axis position is attained

Table 5 General Movement Commands – Single axis and combination. (cont.)

FIG 5b

		Command	Remarks
Interpolation	linear	LIPO (<i>Axis index₁</i> , <i>Axis index₂</i> , <i>Axis index_{3opt}</i> , <i>TYPE_{opt}</i> (<i>End position₁</i>), <i>TYPE_{opt}</i> (<i>End position₂</i>), <i>TYPE_{opt}</i> (<i>End position_{3opt}</i>), <i>Speed₁</i>)	- linear interpolation with a maximum of 3 axes - <i>TYPE</i> : position attribute
(Interpolation)	clockwise (positive)	CIP0 (<i>Axis index₁</i> , <i>Axis index₂</i> , <i>TYPE_{opt}</i> (<i>End position₁</i>), <i>TYPE_{opt}</i> (<i>End position₂</i>), <i>Radius</i> , <i>Speed</i>)	
(Interpolation)	counter-clockwise (negative)	CIPON (<i>Axis index₁</i> , <i>Axis index₂</i> , <i>TYPE_{opt}</i> (<i>End position₁</i>), <i>TYPE_{opt}</i> (<i>End position₂</i>), <i>Radius</i> , <i>Speed</i>)	

Table 6: Interpolation movements

FIG 6

Movement		Command	Remarks
Master Re-set		SEMASTER (<i>Name of combination, axis index</i>)	- axis index indicated becomes the master for that <i>name of combination</i> - re-setting is also possible while the combination is in motion
Manipulation of combination	disabling the combination	DISABLE (<i>name of combination</i>)	- all axes in the combination can be operated separately
	restoring the combination	RESTORE (<i>name of combination</i>)	- restores the most recently active combination configuration
Synchronization movements	synchronizing	SYNCON (<i>name of combination, slave index</i>)	- synchronizes a DEFGear axis with a moving master axis at maximum speed (system variable)
		SYNCONT (<i>name of combination, slave index, time</i>)	- synchronizes a DEFGear axis with a moving master axis at a given time (implies acceleration)
		SYNCONP (<i>name of combination, slave index, profile name</i>)	- synchronizes a DEFCAM axis with an entry profile in the combination
	de-synchronizing	SYNCOFF (<i>name of combination, slave index</i>)	- removes a DEFGear axis at maximum acceleration (system variable) from the combination - decoupled axes can be operated separately
		SYNCOFFT (<i>name of combination, slave index, time</i>)	- removes a DEFGear axis at a given time
		SYCOFFP (<i>name of combination, slave index, profile name</i>)	- removes a DEFCAM axis with an exit profile

Table 7: Movement Commands for the master/slave combination

FIG 7a

Correction Movements	one-time corrective movement on the slave axis	SHIFT (<i>axis index, position, transition profile</i>)	<ul style="list-style-type: none"> - accelerating or slowing a single axis or the master of an axis combination in order to effect a position shift in the shortest way (Error! Reference could not be located.) - print-mark synchronization is programmable in conjunction with the CHECKPOS function
	correction of master position	REDEF_POS (<i>axis index, TYPE_{opt} (position)</i>)	<ul style="list-style-type: none"> - the current or nominal position of an axis is defined without movement to a new absolute position - redefinition possible during movement as well - only the master position can be redefined during movement of a combination - technology: tape-marking synchronization - TYPE_{opt}: nominal or current position
	deleting correction	DELETE (<i>axis index, correction type</i>)	<ul style="list-style-type: none"> - all corrections to the axis named (<i>axis index</i>) are deleted
Rest Cycle	rest at start of cycle	REST (<i>name of combination, slave index, n</i>)	<ul style="list-style-type: none"> - slave axis rests at start of cycle for <i>n</i> cycles - <i>n</i> is of the type INT
	rest at a defined master position	REST_ON_POS (<i>name of combination, slave index, n, position</i>)	<ul style="list-style-type: none"> - command has the same effect as REST if position is not indicated
Insert Cycle	insert at start of cycle	INSERT (<i>name of combination, slave index, n</i>)	<ul style="list-style-type: none"> - slave axis inserted at start of cycle for <i>n</i> cycles
	insert at defined master position	INSERT_ON_POS (<i>name of combination, slave index, n, position</i>)	<ul style="list-style-type: none"> - command has the same effect as INSERT if position is not indicated - if rest has been previously programmed, the same position must be used

Table 7: Movement Commands for the Master/Slave Combination (cont.)

FIG 7b

	Declaration	Remarks
Master/slave combination (position-controlled)	<i>Name of combination:</i> DEFCAM:=Axis index ₁ , Axis index ₂ , ..., Axis index _n	<ul style="list-style-type: none"> • master axis is the first axis indicated in the declaration (Axis index₁) • all profile types are allowed in this combination
Master/slave combination (speed-controlled drive combination)	<i>Name of combination:</i> DEFGEAR:=Axis index ₁ , Axis index ₂ , ..., Axis index _n	<ul style="list-style-type: none"> • combination with rpm synchronization • only the type GPROFIL is allowed in the DEFGEAR combination • electronic transmission also possible via a DEFCAM combination (position-controlled)
Geometric combination (parallel axes in Cartesian coordinate system)	<i>Name of combination:</i> DEFGEO:=Axis index ₁ , Axis index ₂ , Axis index _{3opt}	<ul style="list-style-type: none"> • interpolation movement possible only with the axes declared in DEFGEO

Table 8: Definition of an axis combination

FIG 8

Definition type	Profile declaration	Remarks
tabular	<i>Profile name: TPROFIL (Variable_i, Tolerance_{opt})</i>	<ul style="list-style-type: none"> - <i>Variable</i> is a field defined in the declarations part - when one or more TPROFIL's are used in a combination, a TPROFILE must also be defined as a reference for the master (typically, a value field with constant division) - TPROFIL axes in a combination must have the same field dimensions
closed (complete cycle)	<i>Profile name: FPROFIL (Movement function_i, Tolerance_{opt})</i>	<ul style="list-style-type: none"> - also enables the definition of an electronic transmission (movement function = P1)
constant ratio	<i>Profile name: GPROFIL (Master speed_i, TYPE_{opt}(slave speed)). Profile name: GPROFIL (Master position_i, TYPE_{opt}(slave position))</i>	<ul style="list-style-type: none"> - programming a broken rational transmission ratio - the combination type of the axis determines whether the movement profile will be executed with rpm-synchronization or angle-synchronization - Type: direction attribute indicates the direction in which the slave axis is to follow the master axis
individually	<i>Profile name: SPROFIL {0..number}:= (Master_Min_i, Master_Max_i, movement function_i, Tolerance_{opt}). (Master_Min₂, Master_Max₂, movement function₂, Tolerance_{opt}). ... (Master_Min_m, Master_Max_m, movement function_m, Tolerance_{opt}).</i>	<ul style="list-style-type: none"> - non-closed master interval is permitted - undefined ranges are replaced with the movement function PO (stop) - individual profile shifts can be programmed

Table 9: Profile declaration

FIG 9

Movement Attributes		
Position Attributes	absolute (linear or round axis)	<i>Position</i> <i>A(Position)</i>
	incremental (linear or round axis)	<i>I(Position)</i>
	absolute in the negative direction (round axis)	<i>RN(Position)</i>
	absolute in the positive direction (round axis)	<i>RP(Position)</i>
	move along the shortest path to the absolute position (round axis SP – shortest path)	<i>RSP(Position)</i>
	nominal position	<i>COM(Position)</i>
	current position	<i>CUR(Position)</i>
Direction Attributes	movement in the positive direction <i>Speed</i> is always an absolute value	<i>Speed</i> or <i>P(Speed)</i>
	movement in the negative direction	<i>N(Speed)</i>
	nominal speed	<i>COM(Speed)</i>
	current speed	<i>CUR(Speed)</i>
Selection of the transition profile for an axis	trapezoid profile (limited acceleration)	<i>DYNPROF(Axis index 1)</i>
	jerk-limited	<i>DYNPROF(Axis index, 2)</i>
	parabolic	<i>DYNPROF(Axis index, 3)</i>

Table 10: Movement Attributes

FIG 10

Movement functions	s=slave position ϕ =master position or time base	Function attribute (parameter list)
stop	$s = \text{Value}_2$	$P0(\text{Value}_2)$
constant transformation	$s = \text{Value}_2 * \phi + \text{Value}_1$	$P1(\text{Value}_2, \text{Value}_{1opt})$
2 nd -degree polynomial	$s = \text{Value}_3 * \phi^2 + \text{Value}_2 * \phi + \text{Value}_1$	$P2(\text{Value}_3, \text{Value}_{2opt}, \text{Value}_{1opt})$
3 rd -degree polynomial	$s = \text{Value}_4 * \phi^3 + \text{Value}_3 * \phi^2 + \text{Value}_2 * \phi + \text{Value}_1$	$P3(\text{Value}_4, \text{Value}_{3opt}, \text{Value}_{2opt}, \text{Value}_{1opt})$
4 th -degree polynomial	$s = \text{Value}_5 * \phi^4 + \text{Value}_4 * \phi^3 + \text{Value}_3 * \phi^2 + \text{Value}_2 * \phi + \text{Value}_1$	$P4(\text{Value}_5, \text{Value}_{4opt}, \text{Value}_{3opt}, \text{Value}_{2opt}, \text{Value}_{1opt})$
5 th -degree polynomial	$s = \text{Value}_6 * \phi^5 + \text{Value}_5 * \phi^4 + \text{Value}_4 * \phi^3 + \text{Value}_3 * \phi^2 + \text{Value}_2 * \phi + \text{Value}_1$	$P5(\text{Value}_6, \text{Value}_{5opt}, \text{Value}_{4opt}, \text{Value}_{3opt}, \text{Value}_{2opt}, \text{Value}_{1opt})$
simple sinusoidal line	$s = 1/2 [1 - \cos(\text{Value } \phi \pi)]$	$S0(\text{Value})$
inclined sinusoidal line	$s = \text{Value}_1 \phi - 1/2 \pi [1 - \sin(\text{Value}_2 \phi 2\pi)]$	$S1(\text{Value}_2, \text{Value}_1)$

Table: Movement functions

FIG 11

Declaration direction	Declaration	Remarks/References
Configuration	CONFIGURATION <i>Name</i> ; ... END CONFIGURATION	- corresponds to the entire system
Global variable	VAR_GLOBAL ... END VAR	- the declaration of global variables for a resource requires connection to a module variable
Resource	RESOURCE: <i>Name</i> ; ON <i>Hardware_ID</i> END RESOURCE	- a resource combines software modules that runs under shared hardware
Module	DEFMODUL <i>Name</i> ; ON <i>Module_Designation</i> <i>modulvar</i> ; <i>resourcevar</i> ; <i>modulvar</i> ; <i>direct address</i> ; ... END_MODULE	- the identification symbol ON is used to establish the module type (<i>Module_Designation</i>) at the logical level - the development system includes a description file that assigns a functionally structured software module to each <i>Module_Designation</i> - within the declarations body of modules, module variables are linked with operating resources (direct addressing) and global variables for the resource or configuration

Table 12: Configuration elements

FIG 12

Declaration	General declaration
Global variable for a resource	VAR_GLOBAL <i>Name</i> ; <i>Module name</i> ; <i>variable name</i> ; <i>type</i> ; END VAR
Global variable for configuration	VAR_GLOBAL <i>Name</i> ; <i>resource name</i> ; <i>module name</i> ; <i>variable name</i> ; <i>type</i> ; END VAR

Table 12: Declaration of global variables

FIG 13

General declaration	Remarks
VAR_ACCESS <i>Name: resource name. module name. variable name;</i> <i>type: access;</i> END VAR	- access to output variables of a module - <i>type</i> : elementary or derived data type - <i>access</i> : READ_WRITE or READ_ONLY
VAR_ACCESS <i>Name: resource name. variable name; type: access;</i> END VAR	- access to global variables of a resource
VAR_ACCESS <i>Name: resource name. module name. % log. storage location; type: access;</i> END VAR	- access to directly displayed variables - <i>log. storage location</i>

Table 14: Declaration of access paths

FIG 14

Communication type	Functional module call-up	Remarks
Device status	<i>status :=</i> STATUS (<i>device</i>)	- the status of the designated device (<i>device</i>) is made available to a program on request - communication partner is indicated via <i>device</i> - the <i>status</i> is returned as a value of the type: INT
Reading data	<i>value :=</i> READ (<i>variable name, device</i>)	- a program requests data - access can be controlled by the module from which the data are read - <i>value</i> is a local variable that is assigned the content of the read variable, and must be the same type as <i>variable designation</i>
Writing data	WRITE (<i>variable name, value, device</i>)	- the values are written from a program into the indicated variable of the device - <i>value</i> must have the same data type as <i>variable name</i>
Programmed notification (cannot be acknowledged)	NOTIFY (<i>event, message, device</i>)	- when the defined event occurs (<i>event</i>), messages (<i>message</i>) can be issued to the indicated device (<i>device</i>)
(can be acknowledged)	ALARM (<i>event, message, device, acknowledgment</i>)	- the message issued must be acknowledged (<i>acknowledgment</i>)

Table 15: Communication function

FIG 15

DRAWINGS, Page 17

Number: DE 197 40 550 A1

Int. Cl.⁶: G 05 B 19/04

Date of publication: April 16, 1998

[labels:]

Pile beam

Warp beam

Heald shafts

Batten

Gripper mechanism

Pile wire feed

Pile wire transverse transport

Pile wire removal

Textile removal

Textile storage

Fig 16

DRAWINGS, Page 18

Number: DE 197 40 550 A1

Int. Cl.⁸: G 05 B 19/04

Date of publication: April 16, 1998

[labels:]

Batten

Gripper, left

Gripper, right

Shaft 1

Pile wire beam

Shaft 2

Filling warp

Fig 17a

DRAWINGS, Page 19

Number: DE 197 40 550 A1

Int. Cl.⁶: G 05 B 19/04

Date of publication: April 16, 1998

[labels:]

Shaft 3

Binding warp

Cutting/Clamping device

Pile wire feed

Pile wire removal

Pile wire transverse transport

DRAWINGS, Page 20

Number: DE 197 40 550 A1

Int. Cl.⁶: G 05 B 19/04

Date of publication: April 16, 1998

[labels:]

Multiple-axis module 0

Multiple-axis module 1

Multiple-axis module 2

Single-axis module 3

Single-axis module 4

Input/output module 5

Input / Output

(digital, analogue)

[bottom row, right to left:]

Textile storage

Binding thread storage

Filling thread storage

Pile thread storage

Needle roller

Pile wire transverse transport

Pile wire removal

Pile wire feed

Shaft binding thread

Shaft filling thread

Shaft pile thread

Gripper right

Gripper left

Master axis

Fig 18